

Web 安全第三次作业

陈铭涛

16340024

X.509 证书描述

在密码学中, X.509 是公钥证书的格式标准, 在许多互联网协议中得到应用。

X.509 对公钥证书的格式, 撤销证书列表 (CRLs), 证书验证路径算法等进行了规定。

一个 X.509 证书中包含了其版本号, 证书序列号, 签名算法, 签发者, 证书主体, 有效期, 公钥, 公钥密钥等信息。证书中的信息使用 ASN.1 进行编码, ASN.1 中数据以 tag, 长度, 值的方式进行编码。证书的基本结构在 RFC 5280 中 4.1 节进行了如下的规定 :

```
Certificate ::= SEQUENCE {
    tbsCertificate      TBSCertificate,
    signatureAlgorithm  AlgorithmIdentifier,
    signatureValue      BIT STRING }
(证书主体, 签名算法以及签名值)
```



```
TBSCertificate ::= SEQUENCE {
    version             [0] EXPLICIT Version DEFAULT v1,
    serialNumber        CertificateSerialNumber,
    signature           AlgorithmIdentifier,
    issuer              Name,
    validity            Validity,
    subject             Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID      [1] IMPLICIT UniqueIdentifier OPTIONAL,
                        -- If present, version MUST be v2 or v3
    subjectUniqueID     [2] IMPLICIT UniqueIdentifier OPTIONAL,
                        -- If present, version MUST be v2 or v3
```

```

extensions      [3] EXPLICIT Extensions OPTIONAL
                -- If present, version MUST be v3
    }

```

(证书主体, 包含版本号, 序列号, 签名算法标识, 签发者信息, 有效期, 证书主体, 证书公钥信息, 签发者 ID, 主体 ID 以及扩展段)

```

Version ::= INTEGER { v1(0), v2(1), v3(2) }

```

(证书版本, 值可为 0, 1, 2, 分别代表版本 1, 2, 3)

```

CertificateSerialNumber ::= INTEGER

```

(证书序列号)

```

Validity ::= SEQUENCE {
    notBefore      Time,
    notAfter       Time }

```

(证书有效期, 由开始和结束时间组成)

```

Time ::= CHOICE {
    utcTime        UTCTime,
    generalTime    GeneralizedTime }

```

```

UniqueIdentifier ::= BIT STRING

```

```

SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm        AlgorithmIdentifier,
    subjectPublicKey  BIT STRING }

```

(公钥信息包含公钥算法和公钥数据)

```

Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension

```

```

Extension ::= SEQUENCE {
    extnID      OBJECT IDENTIFIER,
    critical    BOOLEAN DEFAULT FALSE,
    extnValue   OCTET STRING
                -- contains the DER encoding of an ASN.1 value
                -- corresponding to the extension type identified
                -- by extnID
    }

```

```

AlgorithmIdentifier ::= SEQUENCE {
    algorithm      OBJECT IDENTIFIER,
    parameters     ANY DEFINED BY algorithm OPTIONAL }

```

读取 X.509 证书

提交的程序中使用 Go 语言编写读取 X.509 的程序，调用了 Go 语言的 `encoding/asn1` 库进行 ASN.1 编码内容的读取以及 `crypto/x509/pkix` 库进行对签发者与证书主体信息的读取。

在代码中根据上面的 X.509 证书结构定义了如下的结构体用于 ASN.1 读取：

1. `CertificateData`，对应上述结构中的 `Certificate`：

```
type CertificateData struct {
    TBSCertificate      tbsCertificate
    SignatureAlgorithm  AlgorithmIdentifier
    SignatureValue       asn1.BitString
}
```

2. `TbsCertificate`，对应上述结构中的 `TBSCertificate`：

```
type tbsCertificate struct {
    Version          int
    `asn1:"optional,explicit,default:0,tag:0"`
    SerialNumber     *big.Int
    Signature         AlgorithmIdentifier
    Issuer            asn1.RawValue
    Validity          timeSpan
    Subject           asn1.RawValue
    PublicKey         publicKeyInfo
    UniqueId          asn1.BitString `asn1:"optional,tag:1"`
    SubjectUniqueId  asn1.BitString `asn1:"optional,tag:2"`
    Extensions       []extension
    `asn1:"optional,explicit,tag:3"`
}
```

3. `timeSpan`，对应上述结构的 `Validity`：

```
type timeSpan struct {
    NotBefore, NotAfter time.Time
}
```

4. publicKeyInfo, 对应上述的 SubjectPublicKeyInfo:

```
type publicKeyInfo struct {  
    Algorithm AlgorithmIdentifier  
    PublicKey asn1.BitString  
}
```

5. extension, 对应上述结构的 Extension

```
type extension struct {  
    ExtnID      asn1.ObjectIdentifier  
    Critical    bool `asn1:"default:false"`  
    ExtnValue []byte  
}
```

6. AlgorithmIdentifier, 对应上述的 AlgorithmIdentifier:

```
type AlgorithmIdentifier struct {  
    Algorithm  asn1.ObjectIdentifier  
    Parameters asn1.RawValue `asn1:"optional"`  
}
```

程序结构简述

提交的代码中包含三个代码文件 :x509cert/certificate.go 为证书结构体的定义以及对证书中算法的识别函数, 并定义了如下的 certInfo 结构方便其他代码获取证书信息 :

```
type CertInfo struct {  
    Version          int  
    Serial            *big.Int  
    Signature         AlgorithmIdentifier  
    Issuer            IssuerType  
    Validity          timeSpan  
    Subject           IssuerType  
    PublicKey         publicKeyInfo  
    UniqueId          asn1.BitString  
    SubjectUniqueId  asn1.BitString  
    Extensions        []extension  
    SignatureAlgorithm AlgorithmIdentifier
```

```

        SignatureValue    asn1.BitString
    }

type IssuerType struct {
    Country    string
    Province   string
    City       string
    Organization string
    Unit       string
}

```

x509cert/static.go 文件包含了证书读取过程中的一些静态数据如算法的名称与 oid 等。

main.go 文件包含程序的主函数，将根据 pem 格式或 DER 格式读取文件并将证书信息打印至标准输出。

程序编译运行结果

程序使用 go 进行编译，在代码目录下执行命令 `go build -o x509` 即可编译生成命名为 x509 的可执行文件。提交的 bin 文件夹中已包含了使用 go1.11.1 编译获得的 Windows, MacOS 以及 Linux 下的可执行文件。

程序的使用方法为：

```
./x509 [--DER] filename
```

其中若指定了 DER 选项，则程序将会使用 DER 方式读取证书，若未指定则程序会使用 PEM 方式读取证书。

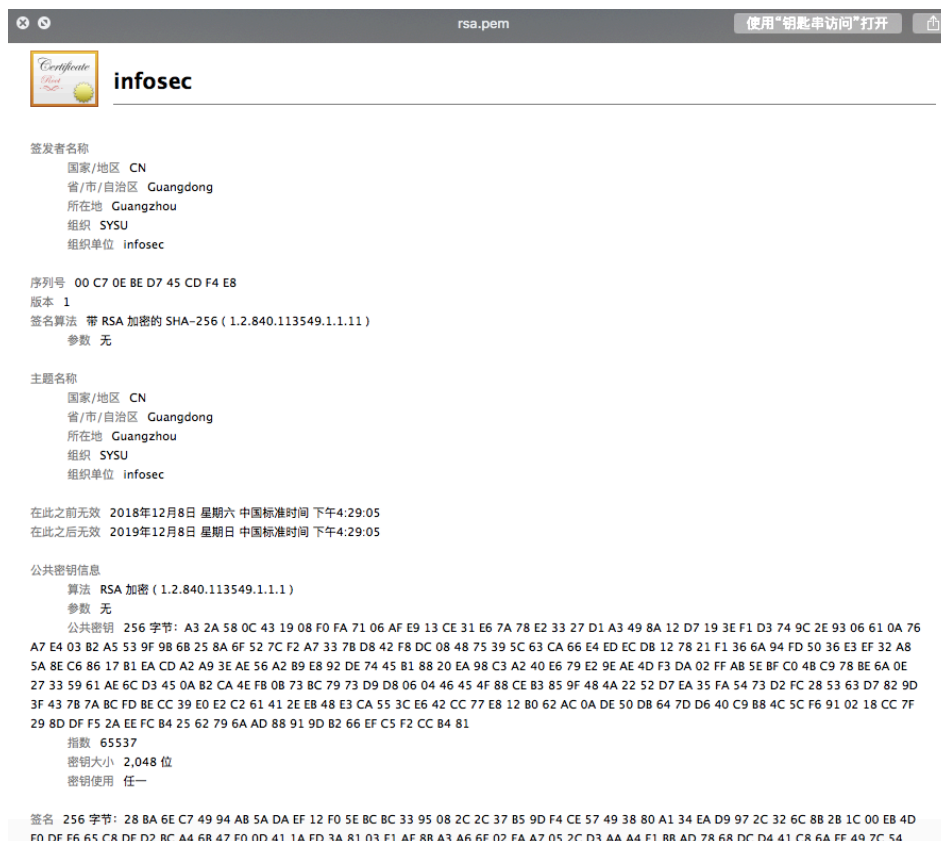
测试运行使用的证书使用 openssl 生成自签名根证书，命令如下：

```
openssl req -new -x509 -days 365 -keyout rsa.key -out rsa.pem
```

命令执行后需要输入证书主体信息：

```
~/Desktop/大三上/web-security/hw/infosec/x509/cert master • openssl req -new -
x509 -days 365 -keyout rsa.key -out rsa.pem
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'rsa.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) []:CN
State or Province Name (full name) []:Guangdong
Locality Name (eg, city) []:Guangzhou
Organization Name (eg, company) []:SYSU
Organizational Unit Name (eg, section) []:infosec
Common Name (eg, fully qualified host name) []:
Email Address []:
```

使用该命令会使用 RSA 算法签名生成有效期为 365 天的根证书，使用系统自带证书工具查看证书信息如下：



使用如下命令将上面命令生成的 pem 证书转换为 DER 证书进行测试：

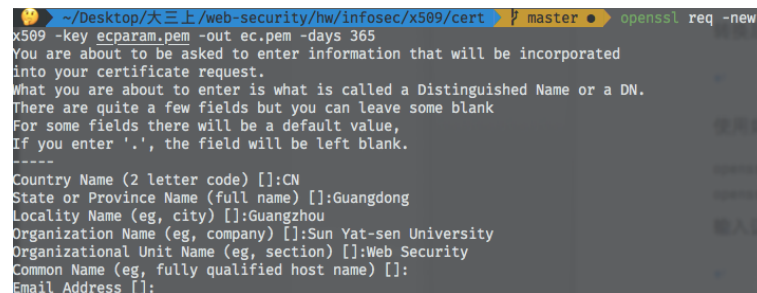
```
openssl x509 -in rsa.pem -outform der -out rsa.crt
```

转换后生成 rsa.crt 证书文件，查看结果与原证书相同。

使用如下命令生成 ECDSA 签名算法的证书用于测试：

```
openssl ecparam -name secp256k1 -genkey -param_enc explicit -out ecparam.pem
openssl req -new -x509 -key ecparam.pem -out ec.pem -days 365
```

输入证书主体信息后生成证书：

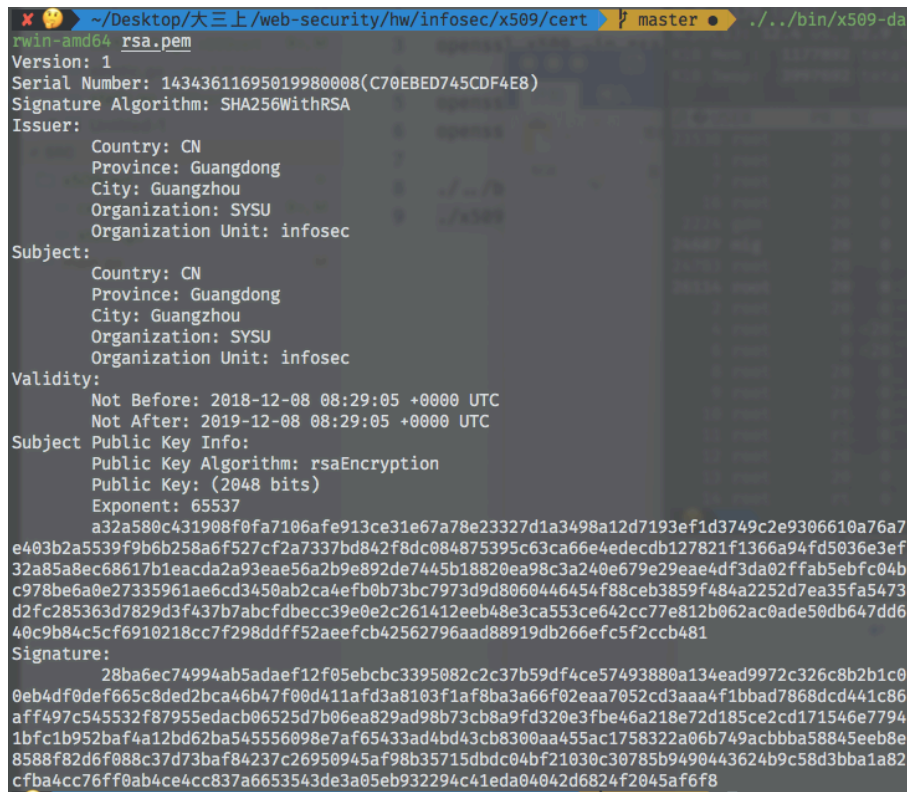


```
~/Desktop/大三上/web-security/hw/infosec/x509/cert master • openssl req -new
x509 -key ecparam.pem -out ec.pem -days 365
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) []:CN
State or Province Name (full name) []:Guangdong
Locality Name (eg, city) []:Guangzhou
Organization Name (eg, company) []:Sun Yat-sen University
Organizational Unit Name (eg, section) []:Web Security
Common Name (eg, fully qualified host name) []:
Email Address []:
```

使用编写的程序在证书目录下执行的效果如图：

1. rsa.pem

```
../../bin/x509-darwin-amd64 rsa.pem
```



```
~/Desktop/大三上/web-security/hw/infosec/x509/cert master • ../../bin/x509-da
rwin-amd64 rsa.pem
Version: 1
Serial Number: 14343611695019980008(C70EBED745CDF4E8)
Signature Algorithm: SHA256WithRSA
Issuer:
  Country: CN
  Province: Guangdong
  City: Guangzhou
  Organization: SYSU
  Organization Unit: infosec
Subject:
  Country: CN
  Province: Guangdong
  City: Guangzhou
  Organization: SYSU
  Organization Unit: infosec
Validity:
  Not Before: 2018-12-08 08:29:05 +0000 UTC
  Not After: 2019-12-08 08:29:05 +0000 UTC
Subject Public Key Info:
  Public Key Algorithm: rsaEncryption
  Public Key: (2048 bits)
  Exponent: 65537
  a32a580c431908f0fa7106afe913ce31e67a78e23327d1a3498a12d7193efd3749c2e9306610a76a7
  e403b2a5539f9b6b258a6f527cf2a7337bd842f8dc084875395c63ca66e4edecdb127821f1366a94fd5036e3ef
  32a85a8ec68617b1eacda2a93eae56a2b9e892de7445b18820ea98c3a240e679e29eae4df3da02ffab5ebfc04b
  c978be6a0e27335961ae6cd3450ab2ca4efb0b73bc7973d9d8060446454f88ceb3859f484a2252d7ea35fa5473
  d2fc285363d7829d3f437b7abcfdbec39e0e2c261412eeb48e3ca553ce642cc77e812b062ac0ade50db647dd6
  40c9b84c5cf6910218cc7f298ddff52aeefcb42562796aad88919db266efc5f2ccb481
Signature:
  28ba6ec74994ab5adaef12f05ebc3c395082c2c37b59df4ce57493880a134ead9972c326c8b2b1c0
  0eb4df0def665c8ded2bca46b47f00d411afd3a8103f1af8ba3a66f02eaa7052cd3aaa4f1bbad7868dcd441c86
  aff497c545532f87955edacb06525d7b06ea829ad98b73cb8a9fd320e3fba46a218e72d185ce2cd171546e7794
  1bfc1b952baf4a12bd62ba545556098e7af65433ad4bd43cb8300aa455ac1758322a06b749acbbba58845eeb8e
  8588f82d6f088c37d73baf84237c26950945af98b35715dbdc04bf21030c30785b9490443624b9c58d3bba1a82
  cfba4cc76ff0ab4ce4cc837a6653543de3a05eb932294c41eda04042d6824f2045af6f8
```

程序显示内容为证书的版本号，序列号，签名算法，签发者，主体，有效期，公钥信息，RSA 公钥数据以及证书签名，与系统工具显示的证书内容相同。

使用 DER 证书文件进行读取的结果如图：

```

~/Desktop/大三上/web-security/hw/infosec/x509/cert  master • ../../bin/x509-darw
in-amd64 --DER rsa.crt
Version: 1
Serial Number: 14343611695019980008(C70EBED745CDF4E8)
Signature Algorithm: SHA256WithRSA
Issuer:
    Country: CN
    Province: Guangdong
    City: Guangzhou
    Organization: SYSU
    Organization Unit: infosec
Subject:
    Country: CN
    Province: Guangdong
    City: Guangzhou
    Organization: SYSU
    Organization Unit: infosec
Validity:
    Not Before: 2018-12-08 08:29:05 +0000 UTC
    Not After: 2019-12-08 08:29:05 +0000 UTC
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    Public Key: (2048 bits)
    Exponent: 65537
    a32a580c431908f0fa7106afe913ce31e67a78e23327d1a3498a12d7193ef1d3749c2e9306610a76a7
e403b2a5539f9b6b258a6f527cf2a7337bd842f8dc084875395c63ca66e4edecdb127821f1366a94fd5036e3ef
32a85a8ec68617b1eacda2a93eae56a2b9e892de7445b18820ea98c3a240e679e29eae4df3da02ffab5ebfc04b
c978be6a0e27335961ae6cd3450ab2ca4efb0b73bc7973d9d8060446454f88ceb3859f484a2252d7ea35fa5473
d2fc285363d7829d3f437b7abcfdbec39e0e2c261412eeb48e3ca553ce642cc77e812b062ac0ade50db647dd6
40c9b84c5cf6910218cc7f298dfff52aeefcb42562796aad88919db266efc5f2ccb481
Signature:
    28ba6ec74994ab5adaef12f05ebcbc3395082c2c37b59df4ce57493880a13ead9972c326c8b2b1c0
0eb4df0def665c8ded2bca46b47f00d411afd3a8103f1af8ba3a66f02eaa7052cd3aaa4f1bbad7868dcd441c86
aff497c545532f87955edac06525d7b06ea829ad98b73cb8a9fd320e3fba6a218e72d185ce2cd171546e7794
1bfc1b952baf4a12bd62ba545556098e7af65433ad4bd43cb8300aa455ac1758322a06b749acbbba58845eeb8e
8588f82d6f088c37d73baf84237c26950945af98b35715dbdc04bf21030c30785b9490443624b9c58d3bba1a82
cfba4cc76ff0ab4ce4cc837a6653543de3a05eb932294c41eda04042d6824f2045af6f8
~/Desktop/大三上/web-security/hw/infosec/x509/cert  master •

```

与 PEM 模式下证书读取获得的结果相同。

读取 ecdsa 加密的证书结果如下：

```

~/Desktop/大三上/web-security/hw/infosec/x509/cert  master • ../../bin/x509-darw
in-amd64 ec.pem
Version: 1
Serial Number: 9424082873066767195(82C911712EFCC35B)
Signature Algorithm: ECDSAWithSHA256
Issuer:
    Country: CN
    Province: Guangdong
    City: Guangzhou
    Organization: Sun Yat-sen University
    Organization Unit: Web Security
Subject:
    Country: CN
    Province: Guangdong
    City: Guangzhou
    Organization: Sun Yat-sen University
    Organization Unit: Web Security
Validity:
    Not Before: 2018-12-08 09:24:22 +0000 UTC
    Not After: 2019-12-08 09:24:22 +0000 UTC
Subject Public Key Info:
    Public Key Algorithm: EcdsaEncryption
Signature:
    30440220759ec987047f38f530d84e927473567c744a37ee04969da2e2a015d4d593fe8022079d3f
dc371d20184ba382317b88796c614e51183efe8c8b60471bf53f0a84190

```

与生成证书时的信息相同。